An approximative inference method for solving ∃∀SO satisfiability problems

Hanne Vlaeminck, Johan Wittocx, Joost Vennekens, Marc Denecker, and Maurice Bruynooghe

Department of Computer Science, K.U. Leuven

Abstract. The fragment $\exists\forall SO(ID)$ of second order logic extended with inductive definitions is expressive, and many interesting problems, such as conformant planning, can be naturally expressed as *finite domain* satisfiability problems of this logic. Such satisfiability problems are computationally hard (Σ_2^P). In this paper, we develop an approximate, sound but incomplete method for solving such problems that transforms a $\exists\forall SO(ID)$ to a $\exists SO(ID)$ problem. The finite domain satisfiability problem for the latter language is in NP and can be handled by several existing solvers. We show that this provides an effective method for solving practically useful problems, such as common examples of conformant planning. We also propose a more complete translation to $\exists SO(FP)$, existential SO extended with nested inductive and coinductive definitions.

1 Introduction

Several declarative problem solving frameworks for solving search problems are based on the computational task of finite model generation. Prominent examples of such frameworks are Answer Set Programming (ASP) [1] and model expansion [10]. In ASP, finite Herbrand models of an answer set program are computed [1]. Model expansion (MX) generalizes Herbrand model generation and aims at computing one or more models of a theory T that expand a finite structure I_0 for a (possibly empty) subset of symbols of T. MX can be applied for arbitrary logics with a model theoretic semantics. In [10], it is shown that MX for first order logic (MX(FO)) is complete for NP problems ("it captures NP"). This property is preserved for rich extensions of FO, such as FO extended with inductive definitions (FO(ID)) [5] and with aggregates. By contrast, disjunctive ASP is complete for Σ_2^P [1]. Formally, $MX(FO)$ is equivalent to the finite domain satisfiability problem for existential second-order logic ($SAT(\exists SO)$). An overview of state-of-the-art ASP and MX(FO($\cdot$)) solvers is in found in [6] (here, FO($\cdot$) refers to arbitrary extensions of FO).

As a running example, consider the following dynamic domain:*a glass may be clean or not, and can be cleaned by the action of wiping.* This is expressed in the following FO theory T_{action}:

$$\begin{array}{rl} & \forall t : (Clean(t+1) \Leftrightarrow Clean(t) \vee Wipe(t)). \\ \wedge & \quad Clean(0) \Leftrightarrow InitiallyClean. \end{array} \tag{1}$$

The bounded planning problem to turn a dirty glass in a clean one in n steps is expressed by the satisfiability problem of the following $\exists SO$ formula in the range $[0 \dots n]$ of time points:

$$\exists Wipe, Clean, InitiallyClean : (T_{action} \wedge \neg InitiallyClean \wedge Clean(n)). \quad (2)$$

For $n > 0$, this formula is indeed satisfied in the suitable interpretation of $0, n, +/1$ and each *witness* W for its satisfiability provides a plan. E.g., wiping at time point 0 will do the job, as is verified by the witness W for which $Wipe^W = \{0\}$ and $Clean^W = \{1, \dots, n\}$.

In this paper, we are not interested in NP, but in the next level Σ_2^P of the polynomial hierarchy. A well-known such problem is finite domain satisfiability for $\exists\forall SO$: satisfaction in finite interpretations is in Σ_2^P for every $\exists\forall SO$ sentence and is Σ_2^P-hard for some such sentences [8]. The same holds for $\exists\forall SO(ID)$. An interesting Σ_2^P problem is that of *conformant planning*, which we discuss in detail in Section 6. Extending our example, suppose that we do not know whether the object is initially clean or dirty, but still want a plan that is guaranteed to make it clean, *no matter what* the initial situation was. This can be formulated as:

$$\exists\, Wipe\, \forall InitiallyClean, Clean : \quad (T_{action} \Rightarrow Clean(n)). \quad (3)$$

In words, we need an assignment to the action $Wipe$ such that the goal is satisfied for every initial situation $InitiallyClean$ and fluent $Clean$ that satisfy the action theory. Note that instead of a conjunction as in (2), in formula (3) we find an implication. Indeed, the condition $T_{action} \wedge Clean(n)$ does not solve the problem as there are many interpretations for the $Clean$ predicate that do not satisfy the action theory, e.g., when $InitiallyClean$ is true and $Clean(0)$ is false.

While Σ_2^P problems can be solved in principle, e.g., by solvers for disjunctive ASP, in practice they are often too hard. In this paper, we present an *approximate* method that consists of reducing a $\exists\forall SO(ID)$ problem to a $\exists SO(ID)$ problem. This method is sound but not complete, in the sense that a witness of the approximating $\exists SO(ID)$ formula is a witness of the $\exists\forall SO$ formula, but not necessarily the other way around. Our method exploits the techniques for *constraint propagation* in FO$(\cdot)$ proposed in [17, 16]. This propagation operates on a three-valued structure that approximates all models of an FO theory and makes it more and more precise. It was shown in [16] that the propagation process can be captured in a formal FO(ID) inductive definition, and we use it here to build the $\exists SO(ID)$ formula. This approach has the advantage that the translation can be automated and that any existing satisfiability solver for $\exists SO(ID)$ or any MX(FO(ID)) solver can be plugged in. Finally, we exploit a result of [11] to develop a more accurate translation of $\exists\forall SO(ID)$ formulas with inductive definitions to $\exists SO(FP)$, $\exists SO$ with nested least and greatest fixpoint definitions. Our method is inspired by interpolation in Logic Programming [2, 12] and approximate query answering in locally closed databases [4].

2 Preliminaries

We assume familiarity with standard first order logic (FO) and second order logic (SO). We define the extensions FO(ID) and FO(FP) of FO, and the corresponding extensions of SO as follows. Given is a vocabulary Σ. A rule (over Σ) is an expression of the form $\forall \bar{x}\ P(\bar{t}) \leftarrow \varphi$ where $P(\bar{t})$ is an atomic formula and φ an FO formula. The symbol $\leftarrow$ is a new connective, called the *definitional implication*, to be distinguished from the FO material implication symbol $\Leftarrow$ (or its more standard inverse $\Rightarrow$). A definition Δ is a finite set of rules. A predicate symbol P in the head of a rule of Δ is called a *defined predicate*; all other predicate and function symbols in Δ are called *open symbols* or the *parameters* of the definition; the set of defined predicates is denoted $Def(\Delta)$, the remaining symbols $Open(\Delta)$. An FO(ID) formula is defined using the standard induction defining an FO formula, augmented with one extra case:

– A definition Δ over Σ is an FO(ID) formula (over Σ)).

FO(ID) formulas are quantified boolean combinations of atoms and definitions. Notice that rule bodies do not contain definitions, that rules only occur inside definitions and are not FO(ID) formulas themselves. The satisfaction relation $I \models \varphi$ of FO(ID) is defined using the standard inductive rules of FO, augmented with one extra rule:

– $I \models \Delta$ if $I = (I|_{Open(\Delta)})^{\Delta}$.

where $(I|_{Open(\Delta)})^{\Delta}$ is the well-founded model of Δ extending the restriction of I to the open symbols of Δ. Here we use the parameterized version of the well-founded semantics that was introduced in the context of deductive databases [15]; it defines *intensional view predicates* (i.e., defined predicates) in terms of a database of *extensional predicates* (i.e., a structure defining open symbols).

The formal notion of a definition as defined here is a faithful syntactic formalisation of informal inductive definitions as used in mathematics [3].

We now define the logic FO(FP). A rule is called *positive* in a set of predicate symbols σ if each predicate of σ has only positive occurrences in rule bodies (i.e., is in the scope of an even number of $\neg$). A *fixpoint definition* is defined inductively as either a *least fixpoint definition* $\lfloor S, \Delta_1, \ldots, \Delta_n \rfloor$ or a *greatest fixpoint definition* $\lceil S, \Delta_1, \ldots, \Delta_n \rceil$, where in both cases S is a set of rules, and $\Delta_1, \ldots, \Delta_n$ are fixpoint definitions. Define the defined predicates $Def(\Delta)$ of a fixpoint definition Δ inductively, as $Def(S) \cup Def(\Delta_1) \cup \cdots \cup Def(\Delta_n)$. We require that predicates of $Def(\Delta)$ have only positive occurrences in rule bodies anywhere in Δ, and also that defined predicates of Δ_i do not occur in Δ_j, $i \neq j$. An FO(FP) formula is defined as in FO with one extra case:

– A fixpoint definition $\mathcal{D}$ over Σ is an FO(FP) formula (over Σ).

With each fixpoint definition $\mathcal{D}$, a monotonic operator $\Gamma_{\mathcal{D}}$ can be associated. This operator is essentially an extension of the standard operator of (unnested) inductive definitions defined by induction on the subdefinition structure of $\mathcal{D}$. We then define:

- $I \models \Delta$ (where Δ is a least fixpoint definition) if I is the least fixpoint of Γ_Δ.
- $I \models \nabla$ (where ∇ is a greatest fixpoint definition) if I is the greatest fixpoint of Γ_∇.

This notion of fixpoint definition is a syntactic variant of the notion of nested least and greatest fixpoint expressions, the difference being that predicate symbols are defined instead of fixpoint expressions denoting relations, and a rule-based syntax is used.

The techniques introduced in the following sections implicitly use concepts of three-valued logic. We assume familiarity with three-valued interpretations and (Kleene's) three-valued truth evaluation. Three-valued interpretations $\mathcal{I}$ assign three-valued relationships to predicate symbols and are used here as approximations of two-valued interpretations I. The precision order $\mathcal{I} \leq_p \mathcal{I}'$ holds if $\mathcal{I}$ and $\mathcal{I}'$ share domain and interpretation of function symbols and $P^{\mathcal{I}}(d_1, \ldots, d_n) = P^{\mathcal{I}'}(d_1, \ldots, d_n)$, for each tuple $(d_1, \ldots, d_n)$ and predicate P/n such that $P^{\mathcal{I}}(d_1, \ldots, d_n) \neq \mathbf{u}$. Two-valued interpretations are maximally precise three-valued interpretations.

We will use a well-known technique to encode three-valued interpretations by two-valued interpretations of an extended language. In particular, let σ be a subvocabulary of Σ such that $\Sigma \setminus \sigma$ contains only predicate symbols. Each three-valued Σ-interpretation $\mathcal{I}$ that is two-valued in every symbol of σ can be encoded as a two-valued I^{tf} of the vocabulary $\Sigma^{tf} = (\sigma \cup \{Q^{ct}, Q^{cf} | Q \in \Sigma \setminus \sigma\}$ such that $\mathcal{I}|_\sigma = I^{tf}|_\sigma$ and such that $(Q^{ct})^{I^{tf}} = \{(d_1, \ldots, d_n) | Q^{\mathcal{I}}(d_1, \ldots, d_n) = \mathbf{t}\}$ and $(Q^{cf})^{I^{tf}} = \{(d_1, \ldots, d_n) | Q^{\mathcal{I}}(d_1, \ldots, d_n) = \mathbf{f}\}$. For a formula φ in negation normal form, we denote by $\varphi^{\text{ct}}_\sigma$ the result of replacing atoms $P(\bar{t})$ by $P^{ct}(\bar{t})$ and negative literals $\neg P(\bar{t})$ by $P^{cf}(\bar{t})$, for every $P \in \Sigma \setminus \sigma$. This encoding has the property that $\varphi^I = \mathbf{t}$ iff $(\varphi^{\text{ct}}_\sigma)^{I^{tf}} = \mathbf{t}$.

3 Propagation for FO

Suppose we have a finite three-valued structure $\mathcal{I}$ that represents some (incomplete) knowledge about the symbols appearing in an FO theory T. We would now like to know the implications of this knowledge. To find this out, we look at the set $\mathcal{M}$ of all models of T that complete this three-valued structure, i.e., $\mathcal{M} = \{M \mid M \models T \text{ and } \mathcal{I} \leq_p M\}$. Given the partial information $\mathcal{I}$, everything that is true in all $M \in \mathcal{M}$ must certainly be true according to T, while everything that is false in all such M must certainly be false according to T. In other words, we can derive from $\mathcal{I}$ the more precise three-valued structure $\mathcal{G}$ that is the greatest lower bound $\text{glb}_{\leq_p}\mathcal{M}$. For instance, let T_{action} be as in (1) and $\mathcal{I}$ the three-valued interpretation that knows that $InitiallyClean$ is $\mathbf{f}$ and $Clean(1)$ is $\mathbf{t}$. In every model of T_{action} that extends this $\mathcal{I}$, it is the case that $Wipe(0)$ holds, and we can figure this out by computing $\mathcal{G}$.

In general, this computation may be too expensive (Δ_2^P) to be of practical use. However, we may still achieve useful results by computing some approximation $\tilde{\mathcal{M}}$ such that $\mathcal{I} \leq_p \tilde{\mathcal{M}} \leq_p \mathcal{G}$. For instance, consider again example (1).

If we know that $\neg InitiallyClean$, we can derive from the second conjunct that also $\neg Clean(0)$, which, according to the first conjunct, implies in turn that the only way to achieve $Clean(1)$ is by $Wipe(0)$. Using this idea, [17] developed a polynomial propagation method to compute such a $\tilde{\mathcal{M}}$. This method was implemented as a C++ program, and is now used in, among others, the grounder of the FO(ID) finite model generator IDP [9].

A recent result in [16] that will prove key to our enterprise here, is that the propagation process can be captured *symbolically* by an FO(ID) definition that defines $\tilde{\mathcal{M}}$. This compilation of an FO theory T into an inductive definition consists of three steps. First, T is rewritten to a theory T' containing only sentences of the form $\forall \bar{x}\ (P(\bar{x}) \Leftrightarrow \varphi)$. Second, these equivalences are split into several sentences of the form $\forall \bar{x}\ (\psi \Rightarrow L[\bar{x}])$, where L is a literal. Finally, these implications are rewritten to rules of an inductive definition.

Definition 1. *An FO sentence φ is in* equivalence normal form *(ENF) if it is of the form $\forall \bar{x}\ (P(\bar{x}) \Leftrightarrow \psi[\bar{x}])$, and ψ is of the form L, $(L_1 \wedge L_2)$, $(L_1 \vee L_2)$, $(\forall v\ L)$ or $(\exists v\ L)$, where L, L_1 and L_2 are literals.*

For the first step, we assume without loss of generality that T is in negation normal form and contains only one formula. The theory T' in ENF is easily obtained from T through a process akin to the Tseitin transformation for propositional logic [14]. Consider the parse-tree of T. For each node $\psi[\bar{x}]$ in this tree that is not a literal, we introduce a new symbol A_ψ/n and add the equivalence $\forall \bar{x}(A_\psi(\bar{x}) \Leftrightarrow \psi')$ where ψ' is obtained from ψ by substituting the Tseitin predicates $A_\phi(\bar{y})$ for non-literal immediate subformulas $\phi[\bar{y}]$ of $\psi[\bar{y}]$.

Proposition 1. *The ENF theory T' is linear in the size of T. Also, M is a model of T iff there exists an expansion M' of M to the vocabulary of T' such that $M' \models T'$ and $M' \models A_T$ where A_T is the Tseitin predicate for T.*

In the second step, we rewrite each ENF formula $\varphi \in T'$ to an equivalent set $T^{\Rightarrow}$ of implications $\mathrm{INF}(\varphi)$ of the form $\forall \bar{x}\ (\psi \Rightarrow L[\bar{x}])$, where L is a literal. The idea is that these implications exhaustively enumerate all the inferences that one could make on the basis of the ENF formula. For instance, if $\varphi = \phi \wedge \psi$, then A_φ implies both φ and ψ, $\varphi \wedge \psi$ implies A_φ, $\neg\varphi$ and $\neg\psi$ both imply $\neg A_\varphi$, $\neg A_\varphi \wedge \varphi$ implies $\neg\psi$, and similarly $\neg A_\varphi \wedge \psi$ implies $\neg\varphi$. Table 1 specifies the corresponding implications for all types of ENF formulas.

The theory $T^{\Rightarrow}$ allows us to characterize the algorithm of [17] in a convenient way: whenever it has inferred the antecedent of such an implication, it infers the consequent. This is reminiscent of Stickel's encoding of clauses in his Prolog Technology Theorem Prover [13]. In the third step, we encode this propagation process explicitly in an inductive definition. Given a theory T over Σ and let $\sigma \subseteq \Sigma$ be a set of symbols on which we have full knowledge in the form of a σ-interpretation I. Assume also that $\bar{Q} = \Sigma \setminus \sigma$ consists of predicate symbols only. We can now use the propagation rules in $T^{\Rightarrow}$ to expand I with three-valued interpretations for the predicates $P \in \bar{Q}$. Recall that three-valued interpretations can be encoded using predicates P^{ct} and P^{cf}. The propagation process in these

φ	INF(φ)
$\forall \bar{x}\ (P(\bar{x}) \Leftrightarrow L[\bar{x}])$	$\forall \bar{x}\ (P(\bar{x}) \Rightarrow L[\bar{x}])$ $\forall \bar{x}\ (\neg L[\bar{x}] \Rightarrow \neg P(\bar{x}))$ $\forall \bar{x}\ (L[\bar{x}] \Rightarrow P(\bar{x}))$ $\forall \bar{x}\ (\neg P(\bar{x}) \Rightarrow \neg L[\bar{x}])$
$\forall \bar{x}\ (P(\bar{x}) \Leftrightarrow \forall y\ L[\bar{x}, y])$	$\forall \bar{x} \forall y\ (P(\bar{x}) \Rightarrow L[\bar{x}, y])$ $\forall \bar{x}\ ((\exists y\ \neg L[\bar{x}, y]) \Rightarrow \neg P(\bar{x}))$ $\forall \bar{x}\ ((\forall y\ L[\bar{x}, y]) \Rightarrow P(\bar{x}))$ $\forall \bar{x} \forall y\ (\neg P(\bar{x}) \wedge (\forall y'\ (y \neq y' \Rightarrow L[\bar{x}, y'])) \Rightarrow \neg L[\bar{x}, y])$
$\forall \bar{x}\ (P(\bar{x}) \Leftrightarrow \exists y\ L[\bar{x}, y])$	$\forall \bar{x} \forall y\ (P(\bar{x}) \wedge (\forall y'\ (y \neq y' \Rightarrow \neg L[\bar{x}, y'])) \Rightarrow L[\bar{x}, y])$ $\forall \bar{x}\ ((\forall y\ \neg L[\bar{x}, y]) \Rightarrow \neg P(\bar{x}))$ $\forall \bar{x}\ ((\exists y\ L[\bar{x}, y]) \Rightarrow P(\bar{x}))$ $\forall \bar{x} \forall y\ (\neg P(\bar{x}) \Rightarrow \neg L[\bar{x}, y])$
$\forall \bar{x} \forall \bar{y} \forall \bar{z}\ (P(\bar{x}, \bar{y}, \bar{z})$ $\Leftrightarrow L_1[\bar{x}, \bar{y}] \wedge L_2[\bar{x}, \bar{z}])$	$\forall \bar{x} \forall \bar{y}((\exists \bar{z}\ P(\bar{x}, \bar{y}, \bar{z})) \Rightarrow L_1[\bar{x}, \bar{y}])$ $\forall \bar{x} \forall \bar{y} \forall \bar{z}\ (\neg L_1[\bar{x}, \bar{y}] \Rightarrow \neg P(\bar{x}, \bar{y}, \bar{z}))$ $\forall \bar{x} \forall \bar{z}((\exists \bar{y}\ P(\bar{x}, \bar{y}, \bar{z})) \Rightarrow L_2[\bar{x}, \bar{z}])$ $\forall \bar{x} \forall \bar{y} \forall \bar{z}\ (\neg L_2[\bar{x}, \bar{z}] \Rightarrow \neg P(\bar{x}, \bar{y}, \bar{z}))$ $\forall \bar{x} \forall \bar{y} \forall \bar{z}\ (L_1[\bar{x}, \bar{y}] \wedge L_2[\bar{x}, \bar{z}] \Rightarrow P(\bar{x}, \bar{y}, \bar{z}))$ $\forall \bar{x} \forall \bar{y}((\exists \bar{z}\ (\neg P(\bar{x}, \bar{y}, \bar{z}) \wedge L_2[\bar{x}, \bar{z}])) \Rightarrow \neg L_1[\bar{x}, \bar{y}])$ $\forall \bar{x} \forall \bar{z}((\exists \bar{y}\ (\neg P(\bar{x}, \bar{y}, \bar{z}) \wedge L_1[\bar{x}, \bar{y}])) \Rightarrow \neg L_2[\bar{x}, \bar{z}])$
$\forall \bar{x} \forall \bar{y} \forall \bar{z}\ (P(\bar{x}, \bar{y}, \bar{z})$ $\Leftrightarrow L_1[\bar{x}, \bar{y}] \vee L_2[\bar{x}, \bar{z}])$	$\forall \bar{x} \forall \bar{y} \forall \bar{z}\ (\neg L_1[\bar{x}, \bar{y}] \wedge \neg L_2[\bar{x}, \bar{z}] \Rightarrow \neg P(\bar{x}, \bar{y}, \bar{z}))$ $\forall \bar{x} \forall \bar{y}((\exists \bar{z}\ (P(\bar{x}, \bar{y}, \bar{z}) \wedge \neg L_2[\bar{x}, \bar{z}])) \Rightarrow L_1[\bar{x}, \bar{y}])$ $\forall \bar{x} \forall \bar{z}((\exists \bar{y}\ (P(\bar{x}, \bar{y}, \bar{z}) \wedge \neg L_1[\bar{x}, \bar{y}])) \Rightarrow L_2[\bar{x}, \bar{z}])$ $\forall \bar{x} \forall \bar{y} \forall \bar{z}\ (L_1[\bar{x}, \bar{y}] \Rightarrow P(\bar{x}, \bar{y}, \bar{z}))$ $\forall \bar{x} \forall \bar{y}((\exists \bar{z}\ \neg P(\bar{x}, \bar{y}, \bar{z})) \Rightarrow \neg L_1[\bar{x}, \bar{y}])$ $\forall \bar{x} \forall \bar{y} \forall \bar{z}\ (L_2[\bar{x}, \bar{z}] \Rightarrow P(\bar{x}, \bar{y}, \bar{z}))$ $\forall \bar{x} \forall \bar{z}((\exists \bar{y}\ \neg P(\bar{x}, \bar{y}, \bar{z})) \Rightarrow \neg L_2[\bar{x}, \bar{z}])$

Table 1. The implications INF(φ) for an ENF formula φ.

predicates is described by an inductive definition of predicates P^{ct} and P^{cf}, for every P appearing in $T^{\Rightarrow}$ but not in σ, i.e., for every symbol of $\bar{Q}$ and also for every Tseitin predicate A_φ.

Definition 2. *For a theory T, set of predicates $\bar{Q}$ such that $\sigma = \Sigma \backslash \bar{Q}$, we define $Approx_\sigma(T)$ as the inductive definition that contains, for every sentence $\forall \bar{x}\ (\psi \Rightarrow L[\bar{x}])$ of $T^{\Rightarrow}$ in which L is a literal of a predicate not in σ, the definitional rule $\forall \bar{x}(L[\bar{x}]^{\mathrm{ct}}_\sigma \leftarrow \psi^{\mathrm{ct}}_\sigma)$.*

Example 1. For the cleaning example, consider the formula $T := T_{action} \Rightarrow Clean(n)$. Take σ to be $\{Wipe, +/1, 0, n\}$. Then $Approx_\sigma(T)$ defines $InitiallyClean^{ct}$, $InitiallyClean^{cf}$, $Clean^{ct}$ and $Clean^{cf}$ as well as A^{ct}_φ and A^{cf} for any introduced Tseitin predicate A_φ, in particular for the Tseitin predicate A_T that is equivalent to the formula $T_{action} \Rightarrow Clean(0)$ and $A_{T_{action}}$ that is equivalent to the conjunction of formulas in the cleaning theory (1). $Approx_\sigma(T)$ then contains amongst others the following definitional rules:

$$\left\{ \begin{array}{ll} A^{ct}_T & \leftarrow A^{cf}_{T_{action}} \vee Clean^{ct}(n). \\ A^{cf}_T & \leftarrow A^{ct}_{T_{action}} \wedge Clean^{cf}(n). \\ A^{ct}_{T_{action}} & \leftarrow A^{ct}_{\varphi_1} \wedge A^{ct}_{\varphi_2} \\ Clean^{ct}(t) & \leftarrow \ldots \\ & \ldots \end{array} \right\},$$

where φ_1 and φ_2 are the two formulas of (1). $Wipe$ is the only open predicate of this definition. In a given σ-interpretation, the definition will compute what

is certainly true and what is certainly false. E.g., when $Wipe(t)$ is false for all t, it will compute that both $Clean^{ct}(t)$ and $Clean^{cf}(t)$ are false for all t.

Proposition 2. *Given T, $\bar{Q}$, $\sigma = \Sigma \setminus \bar{Q}$, a σ-interpretation I, $P \in \bar{Q}$ and a tuple of domain elements $\bar{d}$, the algorithm in [17] will derive the literal $P(\bar{d})$ (respectively $\neg P(\bar{d})$) exactly when $P^{ct}(\bar{d})$ (respectively $P^{cf}(\bar{d})$) holds in the unique model M of the definition $Approx_\sigma(T) \cup \{T^{ct} \leftarrow\}$ expanding I.*

From the correctness of this algorithm, it therefore follows that if $P^{ct}(\bar{d})$ (or $P^{cf}(\bar{d})$) holds in this model M, then $P(\bar{d})$ holds (does not hold) in all models of T that extend I.

4 Approximating ∃∀*SO*-satisfiability problems

Consider the problem whether an $\exists\forall SO$ formula $\exists\bar{P}\forall\bar{Q} : T$ is satisfied in some finite interpretation I of the non-variable symbols of the formula. We call a *witness* for its satisfiability an expansion of I to all existentially quantified variables $\bar{P}$ that satisfies $\forall\bar{Q} : T$. We assume that $\bar{Q}$ consists only of predicate variables (while $\bar{P}$ might include also function variables). We now use the transformation of the previous section to this formula into a stronger $\exists SO(ID)$ formula, i.e., one with less witnesses for $\bar{P}$. Take σ to be the set of all symbols in T except those in $\bar{Q}$.

Definition 3. *For a formula $F = \exists\bar{P}\forall\bar{Q} : T$, we define $\mathcal{APP}(F)$ as the $\exists SO$ formula $\exists\bar{P}\exists\bar{R} : Approx_\sigma(T) \wedge A_T^{ct}$, where $\bar{R} = \{X^{ct}, X^{cf} | X \notin \sigma\}$ (i.e., X a Tseitin symbol A_ϕ or a $Q \in \bar{Q}$) and A_T is the Tseitin symbol for T.*

The intuition here is that for any σ-interpretation I, $Approx_\sigma(T)$ will tell us what the consequences of this choice are, regardless of the value of the universal predicates $\bar{Q}$. If one of these consequences is that the entire FO formula T is true, then we therefore know that I is a witness for the satisfiability of the entire formula F.

Proposition 3. *For each $\exists\forall SO$ formula F of the form $\exists\bar{P}\forall\bar{Q} : T$, $\mathcal{APP}(F)$ is a sound approximation of F, i.e. if $\mathcal{APP}(F)$ is satisfied in interpretation I, then F is satisfied too. Moreover, if I is a witness for the satisfiability of $\mathcal{APP}(F)$, then $I|_\sigma$ is a witness for the satisfiability of F.*

For example, this is the translation of the formula $F = \exists P\forall Q : P \vee Q$:

$$\exists P, Q^{ct}, Q^{cf} : \left\{ \begin{array}{l} T^{ct} \leftarrow P \vee Q^{ct} \\ T^{cf} \leftarrow \neg P \wedge Q^{cf} \\ Q^{ct} \leftarrow T^{ct} \wedge \neg P \\ Q^{cf} \leftarrow T^{cf} \end{array} \right\} \wedge T^{ct}.$$

If we choose P to be true, then the definition forces T^{ct} to be true and T^{cf} to be false. Hence, neither Q^{ct} nor Q^{cf} become true. In other words, choosing P true implies nothing about Q, but it makes the disjunction $T = P \vee Q$ true

for each possible value for Q. This choice for P is therefore a witness for the satisfiability of $\mathcal{APP}(F)$, and it is indeed also a witness for the satisfiability of the original formula $\exists P \forall Q : P \vee Q$.

This approximation method is sound, but for many applications still too incomplete. In particular, it will rarely manage to detect satisfiability of formulas of the form $\exists \bar{P} \forall \bar{Q} : (T_1 \Rightarrow T_2)$. This is because it can only derive that an implication $T_1 \Rightarrow T_2$ holds for all $\bar{Q}$ by either deriving that T_1 is certainly false (i.e., false for all $\bar{Q}$) or that T_2 is certainly true (i.e., true for all $\bar{Q}$). For conformant planning problems (which are of this form, as we will see further), this will never be the case. We will illustrate this with our running example. If we have a look at the definition in example 1, we see that the only way to make A_T^{ct} true is if $A_{T_{action}}^{cf} \vee Clean^{ct}(n)$ is true. However, for a certain choice of Wipe, there are always interpretations for the fluents (e.g. Clean) for which the action theory is satisfied, but also for which it is not satisfied (namely, one of many in which the fluents are simply incorrect for the actions). On the other hand it is also clear that not in all interpretations of the fluents $Clean(n)$ holds. Thus, we will never be able to derive that A_T^{ct} is true. However, we can make our method more complete for problems of the form $\exists \bar{P} \forall \bar{Q} : (T_1 \Rightarrow T_2)$ by postulating the truth of T_1 while checking T_2, as follows.

Definition 4. *For an an $\exists\forall SO$ formula F of the form $\exists \bar{P} \forall \bar{Q} : T_1 \Rightarrow T_2$, we define $\mathcal{APP}^{\Rightarrow}(F)$ as $\exists \bar{P} \exists \bar{R} : \Delta \wedge T_2^{ct})$, where $\Delta = Approx_\sigma(T_1 \Rightarrow T_2) \cup \{T_1^{ct} \leftarrow\}$.*

Note that we add T_1^{ct} as a definitional rule, and T_2^{cf} as a constraint. If we take T_1 to be the trivial formula $\mathbf{t}$, we get back Def. 3 as a special case of this definition. This approximation method is still sound, as the following proposition states.

Proposition 4. *Given a formula F of the form $\exists \bar{P} \forall \bar{Q} : T_1 \Rightarrow T_2$, the $\exists SO(ID)$ formula $\mathcal{APP}^{\Rightarrow}(F)$ is a sound approximation of F, i.e. if $\mathcal{APP}^{\Rightarrow}(F)$ is satisfiable, then F is satisfiable too. Moreover, if I is a witness of the satisfiability of $\mathcal{APP}^{\Rightarrow}(F)$, then $I|_\sigma$ is a witness for the satisfiability of F.*

5 Approximating definitions

In this section, we extend our approximation method to formulas including definitions. We will not consider the general case where definitions may appear at arbitrary locations in a formula, but instead restrict attention to formulas of the form $\exists \bar{P} \forall \bar{Q} : \Delta_1 \wedge \cdots \wedge \Delta_n \wedge \phi \Rightarrow T_2$, where the Δ_i are definitions such that $Def(\Delta_i) \subseteq \bar{Q}$ and ϕ and T_2 are FO formulas. This covers the way in which definitions are typically used: under the assumption that all predicates indeed are what the definitions Δ_i (and the formula ϕ) say they should be, T_2 then states what properties they should satisfy. For instance, in conformant planning, the action theory could include a definition of the fluents in terms of the actions that are performed. By restriction attention to formulas of this form, we avoid the

need for approximation rules that infer that a definition as a whole is certainly true/false.

A first approach is based on the fact that a model of a definition Δ is also a model of the FO completion $compl(\Delta)$. Let us assume w.l.o.g. that each defined predicate P of Δ_i is defined by a single definitional rule $\forall \bar{x}\ P(\bar{x}) \leftarrow \phi$. Then $compl(\Delta)$ consists of all formulas $\forall \bar{x}\ P(\bar{x}) \Leftrightarrow \phi$, for each $P \in Def(\Delta)$. Since definitions occur only negatively in the $\exists\forall SO(ID)$ formula (i.e. in the body of the implication), it is sound to replace each Δ_i by the weaker theory $compl(\Delta_i)$. That is, each witness to $\exists \bar{P} \forall \bar{Q} : compl(\Delta_1) \wedge \cdots \wedge compl(\Delta_n) \wedge \phi \Rightarrow T_2$ is a witness to $\exists \bar{P} \forall \bar{Q} : \Delta_1 \wedge \cdots \wedge \Delta_n \wedge \phi \Rightarrow T_2$. The first formula is $\exists\forall SO$ and we can apply the technique of the previous section.

This method is sound and works fine for non-recursive definitions (where completion is equivalent with FO(ID) semantics) but in case of recursive(=inductive) definitions, it might result in unacceptable loss of precision. For illustration, consider the inductive definition $\{P \leftarrow P\}$. It entails $\neg P$ but this conclusion cannot be derived from its completion. Indeed, the rules in $Approx(P \Leftrightarrow P)$ will obviously fail to derive P^{cf}. As a consequence, the satisfiability of the formula $\forall P : \{P \leftarrow P\} \Rightarrow \neg P$ could not be detected using the above method (since $\forall P : (P \Leftrightarrow P) \Rightarrow \neg P$ is not satisfiable).

As a more complete method, we propose the following. As explained in the preliminaries, for each three-valued interpretation $\mathcal{I}$ of $Open(\Delta)$, the definition Δ has a (three-valued) well-founded model $\mathcal{W}$ extending $\mathcal{I}$. This $\mathcal{W}$ has the interesting property that $\mathcal{W} \leq_p M$, for every model M of Δ such that $\mathcal{I} \leq_p M|_{Open(\Delta)}$. Our aim is now to use this well-founded model $\mathcal{W}$ to make the additional propagations.

In [11], it was shown how the computation of the well-founded model extending a two-valued $Open(\Delta)$-interpretation I can be encoded by a nested fixpoint expression in FO(FP). The following definition extends this to the case of three-valued $Open(\Delta)$-interpretations $\mathcal{I}$ that we need in this context. Let $\sigma \subseteq Open(\Delta)$ such that σ contains all function symbols in Δ. Assume that $\mathcal{I}$ is three-valued only on symbols of $Open(\Delta) \setminus \sigma$.

Definition 5. *For a definition Δ, we define $FP_\sigma(\Delta)$ as $\lfloor \mathcal{R}^{ct}, \lceil \mathcal{R}^{cf} \rceil \rfloor$ where $\mathcal{R}^{ct}$ consists of, the rules*

$$\forall \bar{x}(P^{ct}(\bar{x}) \leftarrow \varphi^{ct}_\sigma)$$

and $\mathcal{R}^{cf}$ consists of the rules

$$\forall \bar{x}(P^{cf}(\bar{x}) \leftarrow (\neg\varphi)^{ct}_\sigma)$$

for every definitional rule $\forall \bar{x}\ P(\bar{x}) \leftarrow \varphi \in \Delta$.

This FO(FP) expression $FP_\sigma(\Delta)$ now does precisely what we want.

Proposition 5. *Given Δ, σ and $\mathcal{I}$ as specified above, let I' be the encoding of $\mathcal{I}$ in terms of the symbols P^{ct}, P^{cf}. Then the unique model of $FP_\sigma(\Delta)$ extending I' encodes the well-founded model of Δ extending $\mathcal{I}$.*

In the case of the definition $\{P \leftarrow P\}$, $FP(\Delta)$ is the following definition: $\lfloor P^{ct} \leftarrow P^{ct}, \lceil P^{cf} \leftarrow P^{cf} \rceil \rfloor$. This definition has a unique model where $P^{ct} = \mathbf{f}$ and $P^{cf} = \mathbf{t}$. This correctly encodes the well-founded model of the original definition, and we see that $FP(\Delta)$ indeed lets us infer that P has to be false.

On the one hand we can now approximate definitions by its completion. Even though we already argued that the approximation of the completion on it's own is not strong enough in the case of recursive definitions, it still does useful propagation. E.g. it allows to propagate information from the defined predicates back to the open predicates. On the other hand we have defined an encoding of the well-founded model of a definition that allows us to minimize predicates. Both of these ways to approximate definitions thus have their own use and we would like to put them together. As defined in the previous section, each approximating definition $Approx_\sigma(F)$ is a positive definiton. This means that we can equally see them as a least fixpoint definition of FO(FP). The following definition then shows how we can put the approximation of the completion of a definition Δ and the encoding of it's well-founded model together.

Definition 6. *The FO(FP) approximation $Approx^{FP}_\sigma(\Delta)$ of an FO(ID) definition Δ is the nested least fixpoint definition $\lfloor Approx_\sigma(Compl(\Delta)) \cup \mathcal{R}', FP_\sigma(\Delta') \rfloor$, where*

- *$Approx_\sigma(Compl(\Delta))$ is the rule set as defined in the previous section,*
- *Δ' is obtained from Δ by replacing all defined predicates P of $Def(\Delta)$ by new symbols P'.*
- *$\mathcal{R}'$ are the rules $P^{ct} \leftarrow P'^{ct}$ and $P^{cf} \leftarrow P'^{cf}$ for every defined predicate P of Δ.*

Note that $FP_\sigma(\Delta')$ is nested in $Approx^{FP}_\sigma(\Delta)$ and is itself a nested definition.

Finally, we now put everything together into an approximation for $\exists\forall SO(ID)$.

Definition 7. *Let F be an $\exists\forall SO(ID)$ formula of the form $\exists\bar{P}\forall\bar{Q} : \Delta_1 \wedge \cdots \wedge \Delta_n \wedge \phi \Rightarrow T_2$, where the Δ_i are definitions and ϕ and T_2 are FO formulas. We then define $\mathcal{APP}^{\Rightarrow}(F)$ as the $\exists SO(FP)$ formula*

$$\exists \bar{P}\, \bar{Q} : \lfloor Approx_\sigma(\phi) \cup Approx_\sigma(T_2) \cup \{A^{ct}_\phi \leftarrow\} \cup \bigcup_{i=1}^{n} Approx^{FP}_\sigma(\Delta_i) \rfloor \wedge T^{ct}_2.$$

Proposition 6. *Given an interpretation I interpreting the non-variable symbols of F. The above defined approximation is sound, i.e. if $\mathcal{APP}^{\Rightarrow}(F)$ is satisfied in I, then F is satisfied in I and moreover, the restriction to σ of a witness of $\mathcal{APP}^{\Rightarrow}(F)$ is a witness of F.*

6 Applications and related work

In the literature, many examples can be found of algorithms that perform some kind of approximate reasoning about the models of a logical theory. Typically, these algorithms, which are specific to the problem at hand, seem to boil down to an instantiation of the general methods presented here. We give some examples.

Conformant Planning In general, a *conformant planning problem* is a planning problem in a non-deterministic domain, where, e.g., the initial state is not fully known. The goal is to come up with a plan that is nevertheless guaranteed to work. This is a hard problem (determining whether there exists a conformant plan of length $\leq k$ is Σ_2^P complete, even if k is assumed to be polynomial in the size of the problem). Therefore, one typically attempts to solve it approximately. [12] starts from a description of the planning problem in the action language $\mathcal{AL}$ and then derives from this an Answer Set Prolog program that searches for solutions in an approximated version of the corresponding transition diagram.

Our method can solve such problems in the following way. Let T_{action} be a theory that defines the fluent predicates $\bar{F}$ in terms of action predicates $\bar{A}$ and initial state predicates $\bar{I_F}$ in the context of a linear time line (possibly a finite interval). This could be an FO theory or an inductive definition as in [5]. Let T_{prec} be a theory describing preconditions $\forall \bar{x} \forall t : A(\bar{x}, t) \Rightarrow \Psi_A[\bar{x}, t]$ of each action predicates A. Finally, let the formula G specify the goal that must be achieved. The problem of conformant planning is then to decide the satisfiability of the following formula:

$$\exists \bar{A} \forall \bar{I_F} \forall \bar{F} : T_{action} \Rightarrow T_{prec} \wedge G$$

In words, there must be a plan ($\exists \bar{A}$), such that no matter how the nondeterministic aspects turn out ($\forall \bar{I}, \bar{F}$), as long as the specification of the effects of the actions is obeyed (T_{action}), the plan will be executable (T_{prec}) and achieve the goal (G). Applying the method of this paper to this formula yields a incomplete algorithm: it may not find all solutions, but if it finds one then that solution is correct. Even though more experiments are needed, preliminary results indicate that this algoritm is comparable to that of [12], both in completeness and runtime.

Querying and reasoning in open databases Approximate methods similar to ours have been used in the context of open databases, databases with CWA [2, 7]. In [4], query answering is considered in the context of databases that are as a whole incomplete, but that nevertheless contain partial, local forms of closed world assumption. The goal is to compute certain answers to queries. Because this task has a high complexity (Δ_2^P), approximate methods are presented which translate an FO query into an approximate FO or FO(FP) query that can be solved directly against the database tables using standard (polynomial) query methods. It is shown that these methods often provide optimal solutions.

The method presented in this paper can provide a similar functionality. Let DB be a set of ground literals, representing an incomplete database. Let Ψ be a background theory: it may contain integrity constraints, view definitions (datalog view programs are FO(ID) definitions), local closed world statements in FO, etc. For a given FO query $Q[\bar{x}]$, the goal is to compute all terms $\bar{t}$ such that $Q[\bar{t}]$ holds in all Herbrand models of $DB \cup \Psi$. The problem of deciding whether a tuple $\bar{t}$ is an answer is the satisfiability problem of $\forall \bar{R}(DB \wedge \Psi \Rightarrow Q[\bar{t}])$, and to this problem our approximate method applies.

While this allows us to decide whether a tuple $\bar{t}$ is a certain answer to the query, it does not yet provide a reasonable method to compute (an approximation of) all such tuples. This can be done as follows. Consider the definition $Approx(DB \wedge \Psi) \cup Approx(Q[\bar{x}])$, consisting of rules describing propagations allowed by the database and rules defining the predicate symbol A_Q^{ct} (A_Q being the Tseitin predicate representing the query $Q[\bar{x}]$). In the unique Herbrand model of this definition, the interpretation of this A_Q^{ct} contains those tuples for which our propagation can derive that they certainly satisfy the query—a sound approximation of the full set of answers. Standard deductive database techniques or techniques from fixpoint logics can be used to compute this relation in polynomial time.

7 Conclusions and future work

Even if a problem is computationally hard in general, specific instances of it might still be solved efficiently. This is why approximate methods are important: they cannot solve every instance, but the instances they can solve, they solve quickly. In computational logic, hard problem arise quite readily. It is therefore not surprising that the literature contains numerous examples of algorithms that perform approximate reasoning tasks for various logical formalisms in various specific contexts. Since many of these algorithms share common ideas, it is a natural question whether they can be seen as instances of some more general method for a more general language.

This paper tries to present such a method. We start from the propagation method for FO($\cdot$) developed in [17] and its symbolic expression in [16] and generalize this to a method for approximating the Σ_2^P-complete $\exists\forall SO(ID)$ satisfiability problem by solving an NP problem. Importantly, this is a syntactic method that transforms the $\exists\forall SO(ID)$ formula into either a $\exists SO(ID)$ or a $\exists SO(FP)$ formula. This affords us the freedom to use any off-the-shelf solver for these languages to perform the approximative reasoning. Moreover, it also makes it significantly easier to update the method by adding (or removing) specific propagations.

In detail, the contributions of this paper are that (1) we have extended the logical representation describing the propagation process to a general method for approximating $SAT(\exists\forall SO)$ problems; (2) we have also added approximations for inductive definitions, using a translation to FO(FP). A final, if somewhat preliminary, contribution is that we have examined how existing approximation methods fit into our general framework. In future work, we hope to extend this analysis, by investigating more thoroughly the relation to these methods, both in terms of efficiency and completeness. Moreover, we are confident that a further literature study will reveal more instances of approximation algorithms that are covered by our results.

References

1. C. Baral. *Knowledge representation, reasoning and declarative problem solving.* Cambridge university press, 2003.
2. Chitta Baral, Michael Gelfond, and Olga Kosheleva. Expanding queries to incomplete databases by interpolating general logic programs. *J. Log. Program.*, 35(3):195–230, 1998.
3. Marc Denecker. The well-founded semantics is the principle of inductive definition. *Logics in Artificial Intelligence, Proceedings of JELIA'98 Schloss Daghstuhl, October 1998*, 1489:1–16, 1998.
4. Marc Denecker, Alvaro Cortés-Calabuig, Maurice Bruynooghe, and Ofer Arieli. Towards a logical reconstruction of a theory for locally closed databases. *ACM Transactions on Database Systems*, 2010. Accepted.
5. Marc Denecker and Eugenia Ternovska. A logic of nonmonotone inductive definitions. *ACM Trans. Comput. Log.*, 9(2), 2008.
6. Marc Denecker, Joost Vennekens, Stephen Bond, Martin Gebser, and Miroslaw Truszczynski. The second answer set programming competition. In *LPNMR*, pages 637–654, 2009.
7. Patrick Doherty, Martin Magnusson, and Andrzej Szalas. Approximate databases: a support tool for approximate reasoning. *Journal of Applied Non-Classical Logics*, 16(1-2):87–118, 2006.
8. N. Immerman. *Descriptive Complexity.* Springer Verlag, 1998.
9. Maarten Mariën, Johan Wittocx, and Marc Denecker. The IDP framework for declarative problem solving. In *Search and Logic: Answer Set Programming and SAT*, pages 19–34, 2006.
10. David G. Mitchell and Eugenia Ternovska. A framework for representing and solving np search problems. In *AAAI*, pages 430–435, 2005.
11. Hou Ping, Broes De Cat, and Marc Denecker. Fo(fd): Extending classical logic with rule-based fixpoint definitions. In *International Conference on Logic Programming (ICLP 2010)*, 2010.
12. Tran Cao Son, Phan Huy Tu, Michael Gelfond, and A. Ricardo Morales. An approximation of action theories of and its application to conformant planning. In *LPNMR*, pages 172–184, 2005.
13. Mark E. Stickel. A prolog technology theorem prover: Implementation by an extended prolog compiler. *J. Autom. Reasoning*, 4(4):353–380, 1988.
14. G. S. Tseitin. On the complexity of derivation in propositional calculus. In A. O. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic II*, volume 8 of *Seminars in Mathematics: Steklov Mathematical Institute*, pages 115–125. Consultants Bureau, New York, 1968.
15. Allen Van Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 47(1):185–221, 1993.
16. Johan Wittocx. *Finite Domain and Symbolic Inference Methods for Extensions of First-Order Logic.* PhD thesis, K.U.Leuven, May 2010.
17. Johan Wittocx, Maarten Mariën, and Marc Denecker. Approximate reasoning in first-order logic theories. In *KR*, pages 103–112, 2008.